임베디드 소프트웨어 최종 과제

12161567 박기수

1. 개요

본 프로그램은 온도 측정과 조도 측정 두 가지 기능을 제공한다. 첫 번째 스위치를 누르면 온도를 측정하고, 두 번째 스위치를 누르면 조도를 측정한다. 측정값은 FND와 LED를 이용해서 출력한다. 온도 측정 모드일 때는 FND에 측정한 온도를 0.5 단위로 출력하고, LED에는 측정한 온도를 2진수로 나타낸다. 조도 측정 모드일 때는 FND에 측정한 조도를 0 ~ 1023 사이의 값으로 출력하고, LED에는 측정한 조도를 8단계로 나누어서 출력한다. 첫 번째 스위치를 누르면 '낮은 도' 음을 출력하고, 두 번째 스위치를 누르면 '높은 도' 음을 출력한다.

2. 작동원리

Mail box, Semaphore, Event flag 총 세 가지의 동기화 메커니즘을 사용했다. Mail box는 측정한 온도 또는 조도를 저장하기 위해 사용했다. Semaphore는 공유 변수인 mode를 바꾸거나 접근하기 위해 사용했다. Event flag는 현재 모드와 관련 없는 task들을 block 시키기 위해 사용했다.

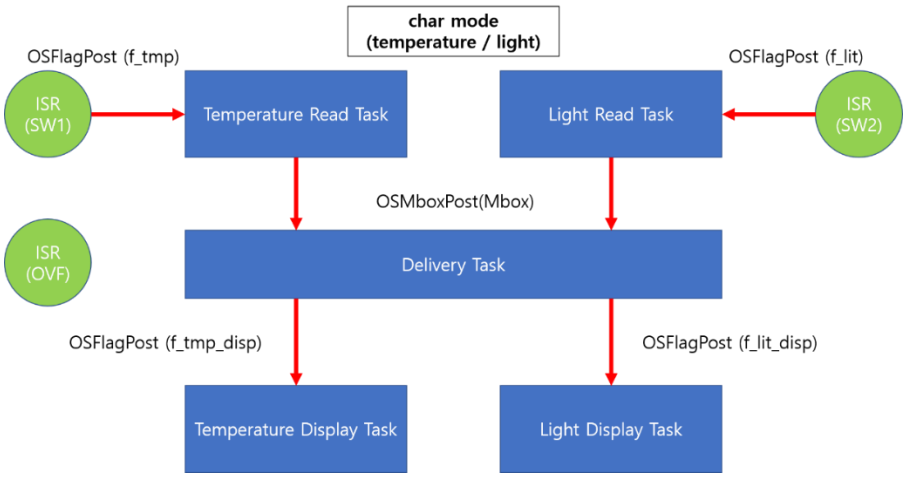

그림 1. 프로그램 진행 과정

1) ISR(SW1), ISR(SW2)

현재 모드를 변경하기 위해 스위치를 누를 때 인터럽트가 호출된다. semaphore를 이용해서 mode를 변경하고, 측정한 값을 저장하는 measured_value를 0으로 초기화한다. semaphore를 빠져나오면 버저음을 출력하기 위해 타이머/카운터2의 인터럽트를 10ms동안 enable 시키고 다시 disable 시킨다. 마지막으로 현재 모드에 해당하는 Read Task에게 event flag를 전달한다.

2) ISR(OVF)

원하는 버저음을 출력하기 위해서 사용한다. 호출될 때마다 state와 PORTB를 토글하고, TCNT2를 원하는 값으로 설정한다. 온도 측정 모드일 경우 TCNT2를 'DO' (낮은 도)로, 조도 측정 모드일 경우에는 TCNT2를 'UDO' (높은 도)로 설정한다.

3) Temperature Read Task, Light Read Task

semaphore를 이용해서 mode를 읽고, 현재 모드와 해당 task의 모드가 일치하지 않으면 OSFlagPend()를 호출해서 해당 task를 block한다. 스위치를 눌렀을 때 호출되는 ISR에서 event flag를 전달받으면 block이 해제된다. 온도 또는 조도를 측정하면 OSMboxPost()를 호출해서 Mail box에 전달한다.

4) Delivery task

semaphore를 이용해서 mode를 읽고, Mail box에 저장된 값을 measured_value에 저장한다. 현재 모드에 따라서 Temperature Display Task 또는 Light Display Task에게 event flag를 전달한다.

5) Temperature Display Task, Light Display Task

semaphore를 이용해서 현재 모드와 측정된 값을 받아온다. 현재 모드와 해당 task의 모드가 일치하지 않으면 OSFlagPend()를 호출해서 해당 task를 block한다. Temperature Display Task는 showTempLed(), showTempFnd() 함수를 호출하고, Light Display Task는 showLightLed(), showLightFnd() 함수를 호출한다.

3. 실행 화면

1) 온도 측정 모드

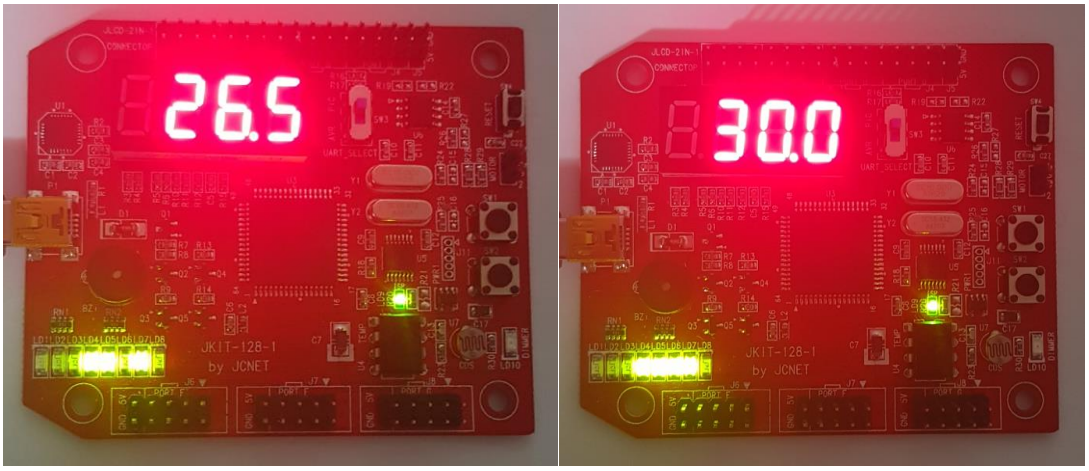

그림 2. 온도 측정 모드

FND에는 0.5 단위로 온도를 출력하고, LED에는 온도를 2진수로 출력한다. 왼쪽 사진 LED의 출력 결과를 2진수로 나타내면 00011010이다. 이를 10진수로 변환하면 26이다. 오른쪽 사진 LED의 출력 결과를 2진수로 나타내면 00011110이다. 이를 10진수로 변환하면 30이다.

2) 조도 측정 모드

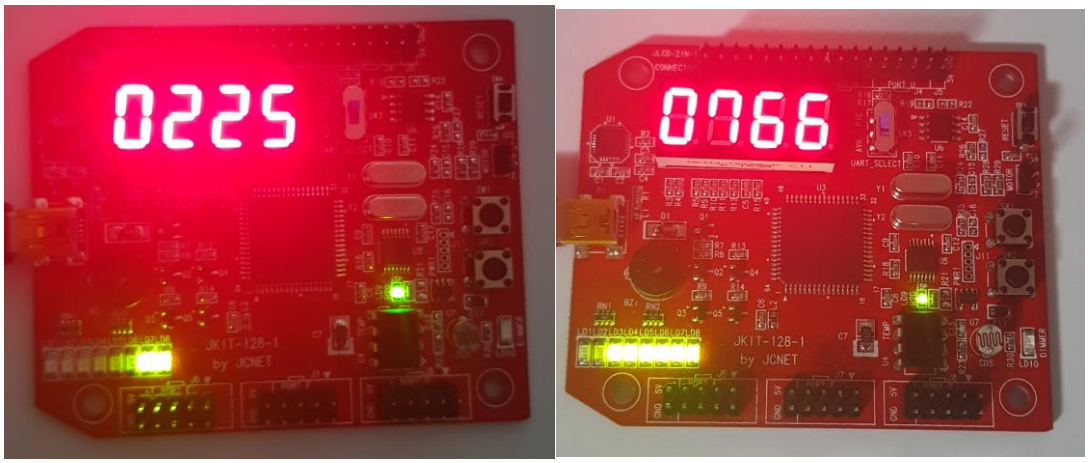

그림 3. 조도 측정 모드

FND에는 측정한 조도를 0 ~ 1023 사이의 값으로 나타낸다. LED에서는 조도를 8단계로 나눠서 출력한다. 조도의 최대값이 1023이므로, LED 하나를 켜기 위해서 128 만큼의 값이 필요하다. 측정값이 0~127이면 LED가 1개, 128~255이면 LED가 2개 켜지고, 896 이상이면 모든 LED가 켜진다. 왼쪽 사진에서는 조도 값이 225이므로 LED 2개가 켜지고, 오른쪽 사진에서는 조도 값이 766이므로 LED 6개가 켜진다.

3) 모드 변환 시 쓰레기 값 출력됨

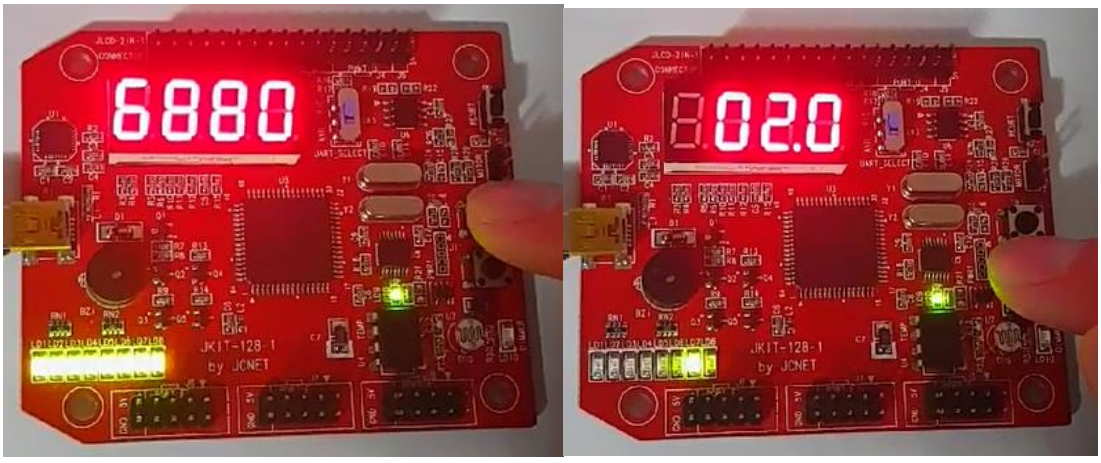

그림 4. 모드 변환 시 쓰레기 값

온도 측정 모드에서 조도 측정 모드로, 조도 측정 모드에서 온도 측정 모드로 변환 시 짧은 시간 동안 쓰레기 값이 출력된다. 쓰레기 값이 출력되는 이유는 Display task를 제외한 모든 task들은 작업을 수행한 뒤 낮은 우선순위 task들의 실행을 위해 delay를 걸어서 block된다. 하지만 Display task는 우선순위가 낮기 때문에 작업을 수행해도 delay를 걸지 않는다. 따라서 모드가 변경되면 Display task는 즉시 모드에 맞게 변경되지만, 값을 측정하는 Read task는 즉시 변경되지 않는다. 왼쪽 사진은 온도 측정 모드에서 조도 측정 모드로 변경했을 때의 상태이다. Display task는 Light Display Task로 변경됐지만, Read task는 아직 Temperature Read task이다. 따라서 측정한 값은 온도지만, 조도 출력 task를 사용했기 때문에 이상한 값이 출력된다. 오른쪽 사진은 조도 측정 모드에서 온도 측정 모드로 변경했을 때의 상태이다. 마찬가지로 Display task는 Temperature Display Task로 변경됐지만 Read task는 아직 Light Read Task이다. 측정한 값은 조도지만, 온도 출력 task를 사용했기 때문에 이상한 값이 출력된다.

4. Pseudocode

<table>
<tr><th>1) ISR(SW1), ISR(SW2)</th><th>2) ISR(OVF)</th></tr>
<tr><td>OSSemPend(sem)
mode <- (변경)
measured_value <- 0
OSSemPost(sem)
(타이머/카운터2 인터럽트 10ms간 활성화)
OSFlagPost(f_temp)</td><td>(state, PORTB 토글)
OSSemPend(sem)
if mode == ‘t’ then TCNT2 <- DO
else if mode == ‘l’ then TCNT2 <- UDO
OSSemPost(sem)</td></tr>
<tr><th>3) Temperature Read Task</th><th>Light Read Task</th></tr>
<tr><td>OSSemPend(sem)
current_mode <- mode
OSSemPost(sem)
if current_mode != ‘t’ then OSFlagPend(f_tmp)
value <- ReadTemperature()
OSMboxPost(Mbox, &value)</td><td>OSSemPend(sem)
current_mode <- mode
OSSemPost(sem)
if current_mode != ‘l’ then OSFlagPend(f_lit)
value <- read_adc()
OSMboxPost(Mbox, &value)</td></tr>
<tr><th colspan="2">4) Delivery task</th></tr>
<tr><td colspan="2">OSSemPend(sem)
current_mode <- mode
OSSemPost(sem)
measured_value <- OSMboxPend(Mbox)
if mode == ‘t’ then OSFlagPost(f_tmp_disp)
else OSFlagPost(f_lit_disp)</td></tr>
</table>

5) Temperature Display Task	Light Display Task

OSSemPend(sem)
current_mode <- mode
value <- measured_value
OSSemPost(sem)
if current_mode != 't'
  then **OSFlagPend(f_tmp_disp)**
showTempLed()
showTempFnd()
	OSSemPend(sem)
	current_mode <- mode
	value <- measured_value
	OSSemPost(sem)
	if current_mode != 'l'
	  then **OSFlagPend(f_lit_disp)**
	showTempLed()
	showTempFnd()